BNB Self-Routing Permutation Network*

Sungchang Lee
Mi Lu

TEXAS A & M University
College Station, TX 77843

Abstract

A self-routing permutation network capable of routing all $n!$ permutations of its n inputs to its n outputs is presented. The network implements the binary radix sorting on the structure of the *generalized baseline network*, a modified model of the original baseline network. The network has $O(N \log^3 N)$ hardware complexity and $O(\log^3 N)$ delay time for N-inputs. The network makes use of the localized bit information instead of the global information in routing procedure. This strategy leads to the reduction of both the hardware and the delay time compared with other comparable networks. The resulting hardware is simple, and has a good regularity.

1 Introduction

A permutation network is a switching device which is capable of routing any of all possible $n!$ permutations of its n inputs to its n outputs simultaneously without any path conflict. The permutation network can be utilized in switching systems and parallel processing systems to provide high communication bandwidth[1, 2]. Well-known permutation networks are crossbar network and cellular interconnection arrays[3, 4]. However, they require too much hardware cost($O(N^2)$ switching nodes) to implement. The Benes network can also realize all permutations using a global routing algorithm[5]. The fastest global routing algorithm known for the Benes network needs $O(\log^2 N)$[1] time when a fully interconnected parallel computer with N processing elements is used[6]. The excessive overhead is rather costly than the network itself.

A self-routing permutation network has been of great interest and much work has been reported. One major advantage of self-routing over global routing is the fast routing time. Nassimi and Sahni[7], and Boppana and Raghavendra[8] showed that rich classes of permutations can be self-routed on the Benes network with simple switch setting strategies. In these algorithms switch setting is determined simply by checking a bit of the destination address. However, these algorithms cannot self-route all permutations.

At the expense of more hardware and proper strategy, a self-routing permutation network can be implemented. In other words, the global routing scheme can be hidden in the hardware so that the network can be regarded as self-routing. As an example, the Batcher's sorting network[9] can be used as a self-routing permutation network. The Batcher's network can be implemented with the hardware complexity of $O(N \log^3 N)$ and the delay time of $O(\log^3 N)$. Also, a self-routing permutation network was presented by Koppelman and Oruç[11]. The network was derived from particular Clos network called the complementary Benes network by modifying the input stage switches, and has $O(\log^3 N)$ delay time and $O(N \log^3 N)$ hardware complexity. The network needs less hardware than the Batcher's network, but it has a longer delay time, complex routing scheme and less regularity of hardware.

In this paper, we present a self-routing permutation network that has less hardware and smaller propagation delay than the forementioned two networks. The network named *baseline nesting baseline(BNB) network* has basic interconnection of the baseline network but each switching stage is replaced by *nested* networks. The nested network also has the interconnection of the baseline network, but the switching devices in each stage are specially designed. The Batcher's network uses compare/swap elements throughout the network. Each of the element compares $logN$-bit to route the inputs properly. The BNB network implements the binary radix sorting algorithm on the *generalized baseline network(GBN)*, a modified model of the original baseline network. Thus, each stage uses only one bit of the destination address for routing. More reduction of the hardware complexity and propagation delay is also obtained by the localization of the routing information processing. The network has a simple routing strategy and a good regularity of the hardware. The analysis shows that the BNB self-routing permutation network needs one third of the hardware needed by the Batcher's network and the delay time is two thirds of that of the Batcher's network.

In section 2, the baseline network is introduced. Also the GBN and some notations are defined. In section

*This research was partially supported by the National Science Foundation under grant no. MIP 8809328.

[1] All logarithms in this paper are to the base 2.

3, a 1-bit slice GBN named *bit-sorter network* is presented, and the structure of BNB self-routing permutation network is described in terms of the GBN. Also, the routing algorithm is presented. In section 4, the structure of a *splitter* which is the core of the self-routing scheme is presented together with the algorithm. The hardware complexity and propagation delay time are analyzed in section 5, and the complexities are compared with other two comparable networks. The conclusion is presented in section 6.

2 The Generalized Baseline Network

In this section, the structure of the baseline network is described and some notations are defined to derive the BNB self-routing permutation network in terms of GBN's.

An N-input baseline network has $m = \log N$ stages of switching elements, from stage-0 to stage-$(m-1)$ beginning with the source side. Let the binary representation of an integer i $(0 \leq i \leq N-1)$ be

$$(b_{m-1}b_{m-2}\cdots b_k b_{k-1}\cdots b_1 b_0),$$

and the 2^k-unshuffle $(1 \leq k \leq m)$ of m-bit number i, $U_k^m(i)$, be

$$U_k^m(i) = (b_{m-1}b_{m-2}\cdots b_k b_0 b_{k-1}\cdots b_1).$$

Let $\mathcal{I}(i,j)$ and $\mathcal{O}(i,j)$, $0 \leq i \leq m-1$, $0 \leq j \leq N-1$, be the j-th input and output of stage-i.

Definition 1 : A *2^k-unshuffle connection of 2^m lines*, U_k^m, between stage-i and stage-$i+1$ is a set of connections such that

$$\mathcal{O}(i,j) = \mathcal{I}(i+1, U_k^m(j)), \quad where \quad 0 \leq j \leq 2^{m-1}. \quad \square$$

The construction of a baseline network can be described by a recursive process[12]. An N-input baseline network consists of an $N \times N$ switching block in stage-0 and two succeeding $\frac{N}{2}$-input baseline networks. The process can be applied to the succeeding blocks recursively. Using the above definition, the connection between stage-i and stage-$(i+1)$ of the baseline network for ($0 \leq i \leq m-2$) is given by,

$$\mathcal{O}(i,j) = \mathcal{I}(i+1, U_{m-i}^m(j)), \quad where \quad 0 \leq j \leq 2^{m-1}.$$

The BNB network structure is based on the baseline network since the interconnection of the baseline network fits the binary radix sorting algorithm naturally. In order to describe the self-routing BNB permutation network, we define some notations in addition to those already given.

Let $SB(i)$ $i = 1, 2, \cdots$ be a 1-bit slice $2^i \times 2^i$ switching box. Note that any 2^i-input and 2^i-output switching device can be considered as switching box $SB(i)$. Also, let $sw(i)$ be a 1-bit slice $2^i \times 2^i$ primitive form of switching box which has 2^{i-1} of 2×2 switches each of which is controlled by an external signal, i.e., a $sw(i)$ needs 2^{i-1} external control signals.

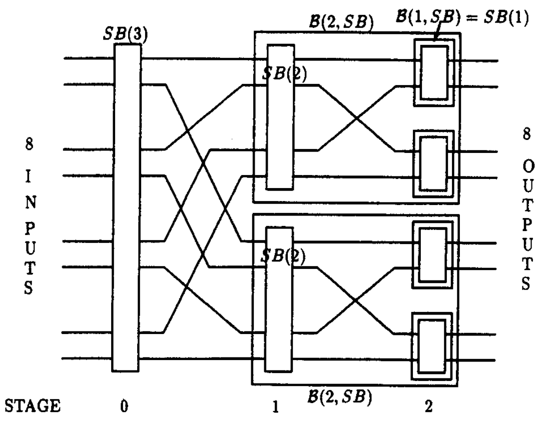

Figure 1: 8-input Generalized Baseline Network, $\mathcal{B}$(3,SB)

Definition 2 : An $N(=2^m)$-input, m-stage *generalized baseline network(GBN)* is a network which has 2^i of $SB(m-i)$'s in each stage-i and 2^{m-i}-unshuffle connections between stage-i and stage-$(i+1)$, $i = 0, 1, \cdots, m-1$. $\square$

GBN may consist of any specific kind of switching box, $SB_k, k = 0, 1, 2, \cdots$. Let $\mathcal{B}_k(m, SB_k)$ denote an 1-bit slice $N(=2^m)$-input, m-stage GBN which consists of switching boxes, $SB_k(l)$, $l = m, m-1, \cdots, 1$. Also, let $\mathcal{B}_k^q(m, SB_k)$ be a q-bit slice 2^m-input GBN of which the k-th slice has 2^i of $SB_k(m-i)$'s in stage-i. Note that each slice of $\mathcal{B}_k^q(m, SB_k)$ may have a different switching box as primitive switching element though they have exactly the same structures. Also, note that a 2^i-input GBN can be considered as a $2^i \times 2^i$ switching box, $SB(i)$. Since a $2^{m-i} \times 2^{m-i}$ switching element of stage-i is connected to the succeeding two 2^{m-i-1}-input GBN's recursively, stage(column)-k of the GBN has 2^k $SB(m-k)$'s. In Fig.1, the notations of 8-input generalized baseline network and the process of the recursive construction is shown. As shown, Fig. 1 can be described by $\mathcal{B}(3, SB)$, which uses switching box, $SB(i)$, and which has 2^3 inputs and 3 stages. As described above, stage-0 has 1 $SB(3)$ and stage-1 has 2^1 $SB(2)$'s and so on. If we construct an N-input generalized baseline network with simple 2×2 switching elements, $sw(2)$'s, then each stage has $\frac{N}{2}$ 2×2 switches, and the baseline network given in [12] is obtained.

3 BNB Self-Routing Permutation Network

3.1 The Bit-Sorter Network

In this subsection, the structure of the *bit-sorter network(BSN)* is described. The BSN will play a major role in the routing to be introduced. The BSN is a GBN which consists of special switching boxes named *splitters* defined below. The splitters generate the *flags* by decoding the inputs. The switches are set by the flags. We first define a particular switching box which is the primitive switching box of the BSN network. The structure will be described in detail and the correctness of the function will be proved in next section.

Let the inputs and outputs of a certain $2^p \times 2^p$ 1-bit slice switching box we are regarding be $s^I(j)$ and $s^O(j)$, $0 \leq j \leq 2^p - 1$. In addition, let $M_e(s^O(j))$, $0 \leq j \leq 2^p - 1$, be the number of the outputs, such that j is even and $s^O(j) = 1$. And, let $M_o(s^O(j))$ be the number of the outputs of which j is odd and $s^O(j) = 1$. The same notation is used for the inputs.

Definition 3 : A *splitter*, $sp(p)$, is a $2^p \times 2^p$ 1-bit slice switching element which can self-route its inputs to the outputs by decoding its inputs, so that $M_e(s^O(j)) = M_o(s^O(j))$, $0 \leq j \leq 2^p - 1$, for $p \geq 2$. If $p = 1$, it routes the inputs such that $s^O(0) = 0$ and $s^O(1) = 1$. □

It is assumed that the number of the inputs, $s^I(j) = 1$ is even for $p \geq 2$. Also, for $p = 1$, it is assumed that one input is 0 and the other is 1. The BSN is defined with the splitter defined above.

Definition 4 : A $2^k \times 2^k$ *bit-sorter network(BSN)* is a 1-bit slice GBN, $\mathcal{B}(k, sp(l))$, $(1 \leq l \leq k)$ in which the *splitters* are the switching elements of the network. □

That is, BSN is a 1-bit slice GBN which uses a specific switching box, splitter, which is defined above as a primitive switching element. Thus, $2^k \times 2^k$ BSN has k stages and 2^l of $sp(k - l)$'s in each stage-l, $0 \leq l \leq k - 1$. Note that, since the splitters self-route their inputs to the outputs, the BSN is a self-routing network.

Theorem 1 *If exactly half of the 2^k inputs are 0'(1's), a bit-sorter network(BSN) routes its inputs to the outputs so that for all outputs, $s^O(j)$,*

$$s^O(j) = \begin{cases} 0 & \textit{if } j \textit{ is even} \\ 1 & \textit{if } j \textit{ is odd.} \end{cases} \quad □$$

Proof : Let us consider a $2^k \times 2^k$ bit-sorter network(BSN). Then, the BSN is a $\mathcal{B}(k, sp(l))$, $0 \leq l \leq k - 1$ by definition. Therefore, it has k stages and 2^l of $sp(k - l)$'s in stage-l ($0 \leq l \leq k - 1$). With the assumption that the number of the inputs with $s^I(j) = 1$ and that of the inputs with $s^I(j) = 0$ are the same, exactly half of the inputs(i.e., 2^{k-1} inputs) are 1(or 0, equivalently). In the first stage of the BSN, the $sp(k)$ receives the inputs and splits them so that $M_e(s^O(j)) = M_o(s^O(j))$ by Definition 3. Since *BSN* is a GBN, $sp(k)$ and the two subsequent $sp(k - 1)$'s are connected by a 2^k-unshuffle connection. Therefore, the even numbered outputs of the $sp(k)$ are connected to the upper $sp(k - 1)$ and the odd numbered outputs are connected to the lower $sp(k - 1)$. Consequently, both of the subsequent two half-size $sp(k - 1)$'s receive 2^{k-2} inputs of $s^I(j) = 1$, equally. If this procedure continues for $(k - 1)$ stages recursively, at the final stage, all the 2^{k-1} $sp(1)$'s, which are 2×2 switching

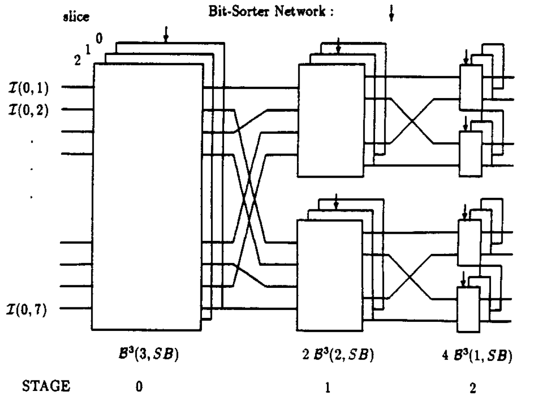

Figure 2: BNB Self-routing Permutation Network, $\mathcal{B}(3, \mathcal{B}^3_k(i, SB_k))$

elements, receive two inputs, of which one is $s^I(j) = 0$ and the other is $s^I(j) = 1$. This final stage switching elements again split their two inputs by the bit, i.e., route the input of 0 bit to the upper output and the other to the lower one by the definition of the splitter. Therefore, all the even numbered outputs of the BSN have 0 and all the odd numbered outputs have 1. □

3.2 The Structure of the Network

In this subsection the structure of the BNB self-routing permutation network is described in terms of GBN's and the bit-sorter networks defined in the previous subsection. Also, the self-routing scheme is presented with the structure. First, we define the structure of the BNB network.

Let $\mathcal{I}(i, j)$ be the input of the BNB self-routing network. Also, assume that the word length of $\mathcal{I}(i, j)$ is $q = m + w$, where m bits are for the destination address and w bits are for the data word. The bits of the $\mathcal{I}(i, j)$, $b^l_{i,j}(\mathcal{I})$ $0 \leq l \leq m - 1$, are for the destination address and $b^0_{i,j}(\mathcal{I})$ is the most significant bit(MSB) of the address. Thus the whole network will be q of 1-bit slice networks. The slices of the networks needed for w-bit data words are only for the general description of the network and they are not involved in the routing of the network. They will follow the routing decided by the bit-sorter network.

Definition 5 : A q-bit slice $2^m(= N)$-input *BNB self-routing permutation network* is a GBN, $\mathcal{B}(m, \mathcal{B}^q_k(i, SB_k))$, $0 \leq k \leq q - 1$, where

$$SB_k(j) = \begin{cases} sw(j) & \text{if } k \neq i \\ sp(j) & \text{if } k = i, \end{cases}$$

and $b^k_{i,j}(\mathcal{I})$'s of $\mathcal{I}(i, j)$'s are input to the k-th slice, $\mathcal{B}_k(i, SB_k)$'s. □

As defined, the structure of the BNB network can be described as 2 level nesting of GBN. The $N(= 2^m)$-input BNB network has the structure of a GBN,

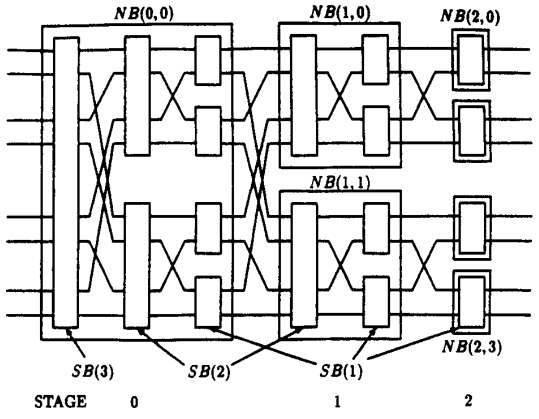

Figure 3: A Profile of BNB Self-routing Permutation Network

$\mathcal{B}(m, SB_a)$, where the switching boxes of the GBN are again the q-slice GBN's, $SB_a(i) = \mathcal{B}_k^q(i, SB_{b_k})$. We refer to the GBN's of different levels as *main network* and *nested network*, respectively. The $N(=2^m)$-input main network has 2^i of $2^{m-i} \times 2^{m-i}$ switching boxes in each stage-i, where the $2^{m-i} \times 2^{m-i}$ switching boxes are q-bit slice 2^{m-i}-input nested GBN's. Each nested network consists of q slices of 1-bit slice GBN. Note that each bit, $b_{i,j}^k(\mathcal{I})$ of $\mathcal{I}(i,j)$ is connected to the k-th slice of the network.

In Fig. 2, $\mathcal{B}(3, \mathcal{B}_k^3(i, SB_k))$, $0 \leq k \leq 2$, is shown. Each input has 3 bits, i.e., the nested networks have three 1-bit slices. The MSB of the input is input to slice-0. Note that the bit-sorter network is the i-th slice of each nested network in stage-i of the main network from Definition 5. Each bit-sorter network in a nested network determines the routing of the whole nested network which it belongs to. The other slices in a nested network follow the routing of the bit-sorter network. The profile of the BNB self-routing network is shown in Fig. 3. The figure shows 1-bit slice of the BNB network out of q-bit slices. From Definition 4 and 5, the bit-sorter network is obtained if the $SB(l)$'s($l = 1, 2, 3$) are replaced by $sp(l)$'s in Fig. 3.

In order to identify the nested GBN's, we denote the l-th nested GBN from the top of the stage-i of the main network by $NB(i,l)$, $0 \leq l \leq 2^i - 1$. Also, let the bit-sorter network(the i-the slice) of the $NB(i,l)$ be $BSN(i,l)$. Then, the main network has $NB(0,0)$ in the first stage, $NB(1,0)$ and $NB(1,1)$ in the second stage and so on, as specified in Fig. 3. Note that $NB(i,l)$ is a 2^{m-i}- input q-bit slice GBN. The 2^{m-i}-input 1-bit slice nested GBN's in the stage-i of the main network are $\mathcal{B}_k(i, SB_k)$'s. Each of them consists of 2^j of $2^{m-i-j} \times 2^{m-i-j}$ switching boxes in the stage-j of the nested network $(0 \leq j \leq m-i-1)$. All the $2^{m-i-j} \times 2^{m-i-j}$ switching boxes are simple switching boxes, $sw(m-i-j)$'s, except the i-th slice of the nested network (i is the stage number of the main network). The i-th slices of nested networks in stage-i of the main network consist of 2^j of $sp(m-i-j)$'s in stage-j of the nested network. This i-th slice of the nested network in stage-i of the main network is a bit-sorter-network(BSN), which is able to self-route its inputs to the outputs. In stage-i of the main network, bit-i, $b_{i,j}^i(\mathcal{I})$, of all the inputs $\mathcal{I}(i,j)$ are connected to the i-th slice of the nested network. As mentioned, the i-th slice of $NB(i,k)$ is a 2^{m-i}-input bit-sorter network.

In summary, the BNB network is a GBN in which the switching boxes are again the q-bit slice nested GBN's of corresponding input sizes. The nested networks are also GBN's which consist of simple switches except the i-th slices. The i-th slices of the nested baseline networks are bit-sorter networks of corresponding input sizes which consist of splitters. That is, only one slice(i-th slice, where i is the stage number of the main network) out of q-slices of the nested network, $\mathcal{B}_k^q(i, SB_k)$ $0 \leq k \leq q-1$, is a bit-sorter network. The bit-sorter network can self-route its inputs to the outputs as defined. All the other $(q-1)$-slices of the nested network consist of 2^j $sw(i-j)$'s. A $sw(i-j)$ consists of 2^{i-j-1} $sw(1)$, i.e., 2×2 simple switches. All the $sw(1)$'s in other slices of the nested network follow the routing of the bit-sorter networks.

3.3 The Self-Routing Algorithm

The permutation capability comes from the proper function of splitters. Since stage-i and stage-$(i+1)$ of the BNB network is connected by a 2^{m-i}-unshuffle connection, U_{m-i}^m, the outputs of a nested network, $NB(i,l)$, in stage-i are connected to two half-size nested networks, $NB(i+1, 2l)$ and $NB(i+1, 2l+1)$, in the subsequent stage-$(i+1)$. That is, the even numbered outputs are connected to the upper half-size nested network and the odd numbered outputs are connected to the lower one. Let $BSN(i,l)$ be the bit-sorter network in the $NB(i,l)$, that is, $BSN(i,l)$ is the i-th slice of $NB(i,l)$. Note that the input, $\mathcal{I}(i,j)$, $0 \leq j \leq N-1(=2^m-1)$, has $q = m+w$ bits each. The m bits, $(b_{i,j}^l(\mathcal{I}),\ 0 \leq l \leq m-1)$, are for the address, in which $b_{i,j}^0(\mathcal{I})$ is the MSB of the address. Therefore, slice-0 of the nested network is for the MSB of the input addresses.

Also recall that all the $sw(1)$'s of $NB(i,l)$'s in stage-i of the main network follow the routings of $sw(1)$'s in the splitters located in the corresponding position in the GBN structure of the $BSN(i,l)$, which is the i-th slice of the $NB(i,l)$. Thus, the routings of the nested network, $NB(i,l)$'s are determined by $BSN(i,l)$'s which are the i-th slice of $NB(i,l)$'s.

Theorem 2 *The BNB network defined can self-route any permutation of the inputs to the outputs.* □

Proof : Recall that a 2^m-input BNB network is a GBN, $\mathcal{B}(m, \mathcal{B}_k^q(i, SB_k))$, $0 \leq i \leq m-1$, $1 \leq k \leq q-1$. Each stage-i of the main GBN has 2^i of q-bit slice 2^{m-i}-input GBN's, $\mathcal{B}_k^q(m-i, SB_k)$, of which the i-th slices are bit-sorter networks(BSN's). Assume the inputs are a permutation of 0 through $N-1(N=2^m)$, then exactly 2^{m-1} inputs will have 0's(1's, equivalently) in bit-0. The $BSN(0,0)$ of $NB(0,0)$ sorts the inputs and routes them to its outputs so that for all outputs $\mathcal{O}(i,j)$, $b^0_{i,j}(\mathcal{O}) = 0$ if j is even, and $b^0_{i,j}(\mathcal{O}) = 1$ if j is odd. As mentioned, all other slices of the $NB(0,0)$ follow the routing of the $BSN(0,0)$. Since the connection between $NB(0,0)$ and succeeding $NB(1,0)$ and $NB(1,1)$ is a 2^m-unshuffle connection, $NB(1,0)$ will receive all the even numbered outputs of $NB(0,0)$ and $NB(1,1)$ will receive all the odd numbered outputs of $NB(0,0)$. That is, $NB(1,0)$ receives all the outputs of $NB(0,0)$ of which bit-0 is 0, and $NB(1,0)$ receives the outputs of which bit-0 is 1. Consequently, stage-1 receives the inputs sorted by bit-0. In stage-1, $BSN(1,0)$ of $NB(1,0)$ and $BSN(1,1)$ of $NB(1,1)$ route the network. Since $BSN(1,0)$ and $BSN(1,1)$ are slice-1 of the nested networks, they take $b^1_{i,j}(\mathcal{I})$'s of the inputs. $BSN(1,0)$ and $BSN(1,1)$ route their inputs so that all the even numbered outputs have $b^1_{i,j}(\mathcal{I}) = 0$ and the odd numbered outputs have $b^1_{i,j}(\mathcal{I}) = 1$. Again, by the 2^{m-1}-unshuffle connection, U^m_{m-1}, between stage-1 and stage-2, $NB(2,0)$ and $NB(2,2)$ receive those inputs having $b^1_{i,j}(\mathcal{I}) = 0$, and $NB(2,1)$ and $NB(2,3)$ receive those inputs having $b^1_{i,j}(\mathcal{I}) = 1$. Of course, from the routing of stage-0, the inputs of $NB(2,0)$ and $NB(2,1)$ have $b^0_{i,j}(\mathcal{I}) = 0$, and the inputs of $NB(2,2)$ and $NB(2,3)$ have $b^0_{i,j}(\mathcal{I}) = 0$. This procedure continues recursively to the final stage. Then at the final stage, i.e., stage-($\log N - 1$) of the main network which has 2^{m-1} of $sp(1)$'s, the inputs are already sorted from the top switch to the bottom one by bit-0 through bit-($m-2$) except bit-($m-1$). Bit-($m-1$) is the least significant bit of the address. The $sp(1)$'s will have one input with $b^{m-1}_{i,j}(\mathcal{I}) = 0$ and the other with $b^{m-1}_{i,j}(\mathcal{I}) = 1$. Finally, by definition of the splitter, they send the inputs with $b^{m-1}_{i,j}(\mathcal{I}) = 0$ to the even numbered outputs and send the inputs with $b^{m-1}_{i,j}(\mathcal{I}) = 1$ to the odd numbered outputs. That is, the inputs(the destination addresses) are routed(sorted) to the outputs where they are destined. Therefore, if the inputs are a permutation of the destination addresses (0 through $N-1$), the BSN self-routes its inputs to the outputs. □

4 The Splitter

In the previous subsection, we showed that if the splitters route their inputs to its outputs so that $Me(s^{\mathcal{O}}(j)) = Me(s^{\mathcal{O}}(j))$ as defined, the BNB network can self-route any permutation of the inputs. The permutation capability basically comes from the idea of splitters and the connections of the network. In this section, we present the switch setting algorithm and the design of the splitter.

A $2^p \times 2^p$ splitter which is the primitive switching element of the bit-sorter network consists of two parts: a 2^p-input *arbiter* and a $sw(p)$.

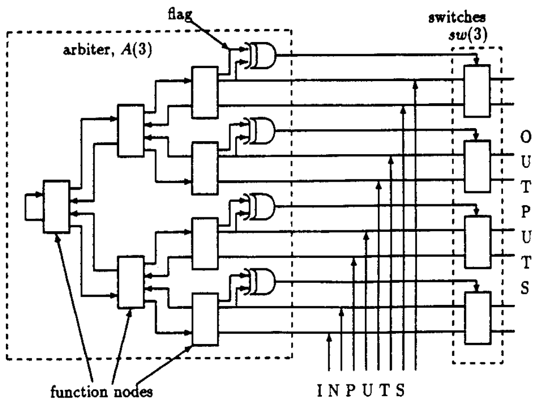

Figure 4: 8-input Splitter, sp(3)

Definition 6 : A 2^p-input *arbiter*, $A(p)$, is a tree-structured function logic which generates *flags* to set the switching box, $sw(p)$, in the $sp(p)$ so that the splitter may work as defined. □

Thus, $sp(p)$ has a $sw(p)$ and an $A(p)$. The structure of an 8-input splitter, $sp(3)$ is shown in Fig. 4. As shown, it consists of an 8-input arbiter, $A(3)$, which has tree structure, and an 8×8 switching element, $sw(3)$.

An arbiter, $A(p)$, in a $sp(p)$ is a 1-bit slice 2^p-input function logic tree. When the bit inputs go up the logic tree, each node of the arbiter sends up state information to its parent to receive flag from its parent. A node generates the flags itself or receive the flag from its parent on condition and sends down the flags to its children. As will be described later, switches are set autonomously using the flags.

We use $sw(1)$'s as the primitive switching device of the design. Thus, a $sw(p)$ has 2^{p-1} 2×2 switches. By the unshuffle connection of the GBN defined, the two outputs of 2×2 switches of a $sp(p)$ are always split into the upper $sp(p-1)$ and the lower $sp(p-1)$ of the subsequent stage. That is because one of the two outputs of a $sw(1)$ is even numbered and the other is odd numbered output and all the even numbered outputs are connected to the upper half-size splitter, $sp(p-1)$ and odd numbered outputs are connected to the lower one by the unshuffle connection.

Let the two outputs of the 2×2 switches, the upper and the lower output, be named as OU and OL, respectively.

Definition 7 : A *type-1 pair* for a 2×2 switch is a pair of the same inputs 0 and 0, or 1 and 1.
A *type-2 pair* refers to the input pair of different bits, 0 and 1, or 1 and 0. □

The switch setting is determined by the inputs and the flag from the arbiter. That is, the input bit is XOR-ed with the flag from the arbiter. This switch setting signal is sent to all other $sw(1)$'s in the corresponding locations of other slices in the same nested network, so that other slices are routed in the same way with BSN. The following switch setting algorithm will split the inputs so that $M_e(s^O(j)) = M_o(s^O(j))$. Consequently, those two succeeding $sp(p-1)$'s have the same number of 0's and 1's.

Let $s^I(j)$ be the input and $f(j)$ be the flag from the arbiter.

Algorithm :

1. Each node of the splitter sends up the XOR of the two input bits to its parent.
2. If the XOR of the two input bits is 0, then it generates the flags itself and sends down 0 to its upper child and 1 to the lower one regardless of the flag from its parent.
3. If the XOR of the two inputs is 1, then it sends down the flag from its parent to both of its children.
4. At the root of the tree, the XOR of the two inputs is echoed as the signal from its parent.
5. The switch setting is

$$s^I(j) = \begin{cases} OU & \text{if } s^I(j) \oplus f(j) = 0 \\ OL & \text{if } s^I(j) \oplus f(j) = 1. \end{cases}$$

Lemma 1 : If the flags of the type-2 pair from the arbiter are 0's (the inputs of the type-2 pair receive the same flags by the algorithm), the input of $s^I(j) = 1$ is always routed to OL and the input of $s^I(j) = 0$ is routed to OU. Also, if the flags are 1's, the input of $s^I(j) = 1$ is always routed to OU and the input of $s^I(j) = 0$ is routed to OL. □

This lemma is obvious from the switch setting strategy.

Theorem 3 *A $2^p \times 2^p$ splitter, $sp(p)$ routes its inputs, $s^I(j)$, $0 \le j \le 2^p - 1$, to its outputs by functioning on the inputs using the described algorithm so that $M_e(s^O(j)) = M_o(s^O(j))$, for $p \le 2$. If $p = 1$, then it routes the inputs such that $s^O(0) = 0$ and $s^O(1) = 1$.* □

Proof : For a 2^p-input splitter, $sp(p)$, it has an $A(p)$ and a $sw(p)$ which consists of 2^{p-1} $sw(1)$'s. The outputs of 2^{p-1} $sw(1)$'s are connected to the succeeding two half-size splitters, $sp(p-1)$'s. Since the connection between them is a 2^p-unshuffle connection of GBN, the even numbered outputs are connected to the upper $sp(p-1)$ and the odd numbered outputs are connected to the lower one. Also, since one of the output of a $sw(1)$ is even numbered output and the other is odd numbered output, the two outputs of the $sw(1)$ are always split to the succeeding upper and lower half-sized splitters, $sp(p-1)$. Thus, as long as the type-1 pair is concerned, the 1's(or 0's) are split to the upper and lower $sp(p-1)$'s equally regardless of the switch settings, *exchange* or *straight*. For the type-2 pair, the XOR of the two inputs is 1, so the node sends up 1 and sends down the flag from its parent. Whenever the XOR of two inputs is 0 at any node, the node sends down 0 to the upper child and 1 to the lower child regardless of the flag from its parent. Since the XOR of type-2 pair is 1, it goes up until it meets another 1 from another type-2 pair to result in 0 on XORing.

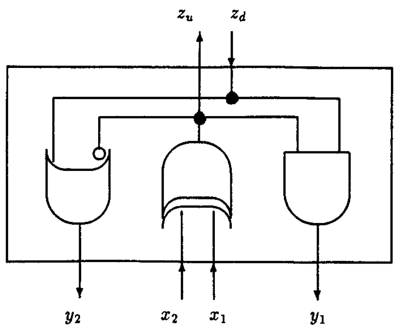

Figure 5: The Schematic of a Function Node of the Arbiter

Then, the node which has 0 as a result of XOR sends down 0 to one child and 1 to the other child. That is, the type-2 pairs are always paired as long as the total number of the type-2 pairs is even, and half of them receive 0 and the other half receive 1 as flags. Consequently, the number of type-2 pairs which receive 0 flags and which receive 1 flags are the same. From Lemma 1, the type-2 pair which receives flags of 0 always routes the input of $s^I(j) = 1$ to OL. Also, the type-2 pair which receives flags of 1 always routes the input of $s^I(j) = 1$ to OU. Therefore, the same number of inputs which have bit 1 are routed to OU's and OL's, i.e., $M_e(s^O(j)) = M_o(s^O(j))$. □

It is assumed that the number of the inputs having $s^I(j) = 1$ is even for $p \le 2$, and the inputs are different(i.e., 0 and 1) for $p = 1$. This assumption is always satisfied since we assume that the inputs are the permutation of the numbers 0 through 2^m. In Fig. 4, only one of the two flags from the arbiter is used for a pair of inputs in switching setting under the assumption of no conflict. However, the other flags and the other inputs can be used to deal with the conflicts if needed in some applications.

The implementation of the schematic logic of the function node is shown in Fig. 5. An arbiter has a tree structure of the function nodes as mentioned. In Fig. 5, x_1 and x_2 are the inputs from the children, and z_u is the information sent up to the parent. z_d is the flag from the parent, and y_1 and y_2 are the flags from the parent. As shown, for the type-1 pair, the node always generates $y_1 = 0$ and $y_2 = 1$ regardless of the flag from the parent. For the type-2 pair, it sends the flag from the parent to the children. All the function nodes of the arbiter are the same. So, the structure is regular and simple. No other strategies are needed to set the switches. The whole BNB self-routing network basically needs only the function nodes shown in Fig. 5 and the 2 x 2 switches, $sw(1)$'s.

5 The Analysis of Complexities

In this section, the hardware complexity and propagation delay are analyzed and they are compared with those of the Batcher's odd-even sorting network and the network presented by Koppelman [11].

5.1 Hardware Complexity

Let $C_{BNB}(N)$ and $C_{NB}(N)$ be the hardware complexity of the N-input BNB self-routing permutation network and N-input nested baseline network, respectively. Then, the hardware complexity of the N-input BNB self-routing permutation network can be expressed by the recurrence equation

$$C_{BNB}(N) = 2 \cdot C_{BNB}(\frac{N}{2}) + C_{NB}(N). \qquad (1)$$

The cost of the P-input nested network is also described by the recurrence equation. Let w be the length of the data word, then we need $\log P$ bits for the destination address and w bits for the data. Thus, $(\log P+w)$ slices of 1-bit slice networks are needed, and one of them is BSN. BSN has an arbiter in addition to the switching elements needed for other slices. As a consequence, the cost of a P-input nested network is

$$C_{NB}(P) = C_{NB,SW1}(P) \cdot (\log P+w) + C_{NB,A}(P), \quad (2)$$

where $C_{NB,SW1}(P)$ is the cost of the switches in an 1-bit slice of a P-input nested network, and $C_{NB,A}(P)$ is the cost of the arbiter in a P-input bit-sorter network slice of P-input nested network. Since P-input nested GBN has $\log P$ stages,

$$C_{NB,SW1}(P) = (\frac{P}{2} \cdot \log P) \cdot C_{SW}, \qquad (3)$$

where, C_{SW} is the cost of a 2×2 switch.

The arbiters have tree structures, so the number of a P-input arbiter has $P-1$ nodes. Therefore, the cost of arbiters in a P-input bit sorter network is described by the recurrence equation

$$\begin{aligned} C_{NB,A}(P) &= C_A(P) + 2 \cdot C_A(\frac{P}{2}) \\ &= (P-1) + 2 \cdot C_A(\frac{P}{2}) \qquad (4) \\ &= (P \log(\frac{P}{2}) - \frac{P}{2} + 1) \cdot C_{FN}, \end{aligned}$$

where C_{FN} is the cost of a node. Note that $A(1)$ does not consist of the function nodes. It is a wiring since the input bit itself is the switch setting signal for $A(1)$.

From equation (1), (3) and (4), the cost of a nested network is given by,

$$\begin{aligned} C_{NB}(P) = &(\frac{P}{2} \cdot \log P(\log P + w)) \cdot C_{SW} \\ &+(P \log(\frac{P}{2}) - \frac{P}{2} + 1) \cdot C_{FN}. \qquad (5) \end{aligned}$$

Finally, from the recurrence equation (1) and (5), the total cost of N-input self-routing BNB permutation network is described by,

$$\begin{aligned} C_{BNB}(N) &= 2 \cdot C_{BNB}(\tfrac{N}{2}) + C_{NB}(N) \qquad (6) \\ &= (\tfrac{N}{6} \log^3 N + \tfrac{N}{4} \log^2 N + \tfrac{N}{12} \log N \\ &\quad + \tfrac{Nw}{4}(\log^2 N + \log N)) \cdot C_{SW} \\ &\quad +(\tfrac{N}{2} \log^2 N - N \log N + N - 1) \cdot C_{FN}. \end{aligned}$$

5.2 Propagation Delay

Since the arbiters are the 1-bit slice function logics, each node does not have logic units like $\log N$-bit adders which may cause propagation delay inside the node.

Let D_{SW} be the delay of a 2×2 switch, and D_{FN} be the delay of a node of an arbiter. The propagation delay, $D_{BNB,SW}$, of the switches in the network is given by the total number of the stages in the network.

$$\begin{aligned} D_{BNB,SW} &= \sum_{k=1}^{\log N} k \cdot D_{SW} \\ &= \frac{1}{2} \log N(\log N + 1) \cdot D_{SW}. \qquad (7) \end{aligned}$$

Since each splitter in the nested network has an arbiter of corresponding size, the total propagation delay of the arbiters in the self-routing BNB permutation network is described by

$$\begin{aligned} D_{BNB,FN} &= 2 \cdot \textstyle\sum_{k=2}^{\log N} \sum_{l=2}^{k} l \cdot D_{FN} \qquad (8) \\ &= (\tfrac{1}{3} \log^3 N + \log^2 N - \tfrac{4}{3} \log N) \cdot D_{FN}. \end{aligned}$$

Note that $A(1)$ is only a wiring of the input bit to switch for the setting. Therefore, the total propagation delay, D_{BNB} of the self-routing BNB permutation network is, from (7) and (8), described by

$$\begin{aligned} D_{BNB} &= (\frac{1}{3} \log^3 N + \log^2 N - \frac{4}{3} \log N) \cdot D_{FN} \\ &\quad +(\frac{1}{2} \log^2 N + \frac{1}{2} \log N) \cdot D_{SW}. \qquad (9) \end{aligned}$$

5.3 Comparison

As forementioned, the Batcher's odd-even sorting network can connect any permutation of its inputs to its outputs without conflict. In this subsection, the hardware and the delay time complexities are compared. The number of comparison elements in the $N(=2^m)$-input Batcher's odd-even sorting network for one-bit input is given by [9],

$$C_{CE,BAT}(N) = \frac{N}{4} \log^2 N - \frac{N}{4} \log N + N - 1. \quad (10)$$

The comparison element of the Batcher's sorting network consists of switches and function logics for comparison. For the $\log(N)$-bit address and w-bit data word length, the total hardware complexity C_{BAT} is

$$\begin{aligned} C_{BAT}(N) = &(\tfrac{N}{4} \log^3 N + \tfrac{N(w-1)}{4} \log^2 N \\ &-(\tfrac{Nw}{4} - N + 1) \log N + (N-1)w) \cdot C_{SW} \\ &+(\tfrac{N}{4} \log^3 N - \tfrac{N}{4} \log^2 N + (N-1) \log N) \cdot C_{FN}. (11) \end{aligned}$$

The delay time of the Batcher's sorting network, D_{BAT} is

$$\begin{aligned} D_{BAT} &= (\frac{1}{2} \log^3 N + \frac{1}{2} \log^2 N) \cdot D_{FN} \\ &\quad +(\frac{1}{2} \log^2 N + \frac{1}{2} \log N) \cdot D_{SW}. \quad (12) \end{aligned}$$

Table 1: Hardware Complexities

Hardware / Network	2×2 Switches	Function Slices	Adder Slices
Batcher	$\frac{N}{4}\log^3 N$	$\frac{N}{4}\log^3 N$	–
Koppelman[11]	$\frac{N}{4}\log^3 N$	$\frac{N}{2}\log^2 N$	$N\log^2 N$
This paper	$\frac{N}{6}\log^3 N$	$\frac{N}{2}\log^2 N$	–

Table 2: Propagation Delay

Network	Delay
Batcher	$\frac{1}{2}\log^3 N + \frac{1}{2}\log^2 N$
Koppelman[11]	$\frac{2}{3}\log^3 N - log^2 N + \frac{1}{3}\log N + 1$
This paper	$\frac{1}{3}\log^3 N + \frac{3}{2}\log^2 N - \frac{5}{6}\log N$

In table 1, the complexities of three comparable networks are shown in terms of 2×2 switches and function logic units. The self-routing permutation network(SRPN) presented by Koppelman, *et. al.* has adder-slices for their *ranking circuit* and function logics to route the inputs according to the routing table in addition to the switches. We compare the complexities assuming that those function and switch slices of the three networks are comparable. As shown in Fig. 5, the function node of the BNB network consists of few gates. Also it is simple and regular.

Table 2 shows the propagation delay of the three networks. As shown, assuming D_{SW}, D_{FN}, C_{SW} and C_{FN} of the three networks are comparable, both the hardware complexity and the delay time of the self-routing BNB permutation network are smaller than those of the Batcher's and Koppelman's. Note that the highest term of the delay time is due to the delay of the function nodes only, and the delay of the function node of this paper is only the delay of one gate. For the lower order terms, the delays of switches and function nodes are added together.

Compared with the Batcher's network, the Koppelman's network(SRPN) [11] saves hardware. It is basically because the network does not consist of compare/swap units which the Batcher's network consists of throughout the network. Instead, it uses ranking circuits and cube networks to route the inputs. The ranking circuit is a tree which consists of four kinds of adder nodes. The switches of the cube network are set for bit sorting according to preset routing rules using the rankings from the ranking circuit and the input bits. Further reductions were obtained in this paper by implementing the binary radix sorting on baseline network structure. In contrast to the efforts to sort bits with global informations, the splitting needs only local bit informations. Each node of splitter needs two bits from its two children and one bit from its parent for decision. At the end of each stage of the main network, the bits are completely split and the interconnection of the baseline network naturally groups the sorted entries. The structure is also simple and regular consequently. All nodes are the same one-bit function logics.

6 Conclusion

In this paper, we have presented the algorithm and the structure of a self-routing permutation network. The hardware cost complexity is $O(N\log^3 N)$ and the propagation delay is $O(\log^3 N)$. Though the complexities are of the same order as those of the Batcher's odd-even sorting network, our network has less hardware and smaller delay time. The analysis shows that the network needs about one third of the hardware of the Batcher's network and the routing delay time is two thirds of that of the Batcher's network by the highest order term comparison. In addition, the routing algorithm is simple, and the hardware has a good regularity.

Acknowledgement

The authors would like to thank Dr. B. Douglass for helpful discussions on these topics.

References

[1] T. Feng, "A survey of interconnection networks," in *IEEE Computer*, Dec. 1981, pp. 12-27.

[2] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. Comput.*, Vol. C-24, pp. 1145-1155, Dec. 1975.

[3] W. H. Kautz *et al.*, "Cellular interconnection arrays," *IEEE Trans. Comput.*, vol. C-17, pp. 443-451, May 1968.

[4] A. Y. Oruç and D. Prakash, "Routing algorithms for cellular interconnection arrays," *IEEE Trans. Compt.*, vol. C-33, pp. 939-942, 1984.

[5] A. Waksman, "A permutation network," *J. Assoc. Compt. for Mach.*, vol. 15, pp. 159-163, 1968.

[6] D. Nassimi and S. Sahni, "Parallel algorithms to set up the Benes permutation network," *IEEE Trans. Compt.*, vol. C-31, pp. 148-154, 1982.

[7] D. Nassimi and S. Sahni, "A self-routing Benes network and parallel permutation algorithms," *IEEE Trans. Compt.*, vol. C-30, pp. 332-340, 1981.

[8] R. Boppana and C. S. Raghavendra,"On self routing in Benes and shuffle exchange networks," in *Proceedings of 1988 International Conference on Parallel Processing*, vol. 1, pp. 196-200.

[9] K. E. Batcher, "Sorting networks and their applications," in *Proc. 1968 Spring Joint Comput. Conf.*, vol. 32, AFIPS, Montvale, NJ, pp. 307-314.

[10] B. Douglass and A. Oruç, "Self-routing and route balancing in connection networks," in *Proceedings of 1990 International Conference on Parallel Processing*, vol. I, pp. 331-337.

[11] D. Koppelman and A. Oruç, "A self-routing permutation network," in *Proceedings of 1989 International Conference on Parallel Processing*, vol. I, pp. 288-295.

[12] C. L. Wu and T. Feng, "On a class of multistage interconnection networks," *IEEE Trans. Comput.*, vol. C-29, pp. 694-702, Aug. 1980.